Comparison between Common Lisp's "setf", "setq" and "set"

Operator	Type	First argument	Targets lexical variables?	Targets data structures (places)?	Modern status
setf	Macro	Unevaluated symbol or access form	**Yes**	**Yes**	**Standard Choice**
setq	Special Operator	Unevaluated symbol only	**Yes**	No	Legacy/Optional
set	Function	Evaluated symbol	No	No	Obsolete/Deprecated

Modern **Common Lisp** style strongly favors "**setf**" for nearly all assignments, rendering "**setq**" mostly optional and "**set**" virtually obsolete.

setf (set field, set find)

The "**setf**" macro is Common Lisp's **generalized assignment operator**. Instead of just taking a variable name, it can take an access form (a "place") and invert it into a mutation form.

Mechanism: It looks at the structure provided, determines how to write to it under the hood, and expands into the appropriate setter logic.

Best used for: All modern Common Lisp assignment tasks, including updating data inside objects, structures, lists, and arrays.

setq (set quoted)

The "**setq**" special operator assigns values **strictly** to **variables**. The name comes from "set quoted" because it automatically quotes its first argument rather than evaluating it.

Mechanism: It expects a literal symbol name as its first argument and modifies its value binding. It can update both local (lexical) and global (special) variables.

Limitation: It cannot handle data structure mutation: *(setq (car x) 5)* is an error.

set

The "**set**" operator is a **standard function**, which means it obeys normal evaluation rules: **all** of it's arguments are evaluated **before** execution.

Mechanism: Because it evaluates its first argument, you must explicitly quote the symbol you want to mutate.

Limitation: As a function, it does not know anything about your lexical environment. It can only see and modify the global, dynamic symbol-value slot of a symbol. Trying to use it on a local variable inside a "**let**" block will modify the global symbol slot **instead** of your local variable.

Modern best practices

Use "setf" by default: Almost all modern Lisp style guides suggest using "**setf**" uniformly for variable modification, as it keeps your code consistent regardless of whether you are modifying a local variable or an array index.

Avoid "set": Do not use "**set**" unless you are doing low-level metaprogramming or modifying the global symbol table directly.

Remember initialization: None of these operators should be used to introduce **new** global variables out of thin air. Always **declare** global variables first using "**defvar**" or "**defparameter**" to ensure proper dynamic binding behavior.